# Functional Programming and the Scala Language

## Lecture 3

**Eugene Zouev**
Innopolis University
Spring Semester 2018

# To Remind:

- FP cornerstones: immutable objects & functions as values
- Scala: object-oriented meets functional; imperative and/or OO and/or functional paradigms
- Function definitions & local functions
- Functions & operators
- Function literals; closures
- Partially-applied functions & currying
- By-name parameters
- Tuples & traits

**Today**:

- Currying & new control structures

# Currying & New Control Structures

Though the language syntax is fixed, it's possible to create **new control structures** using currying feature.

First simple example:

```
def twice(op: Double=>Double, x: Double) = op(op(x))
```

The function repeats the operation two times and returns the result:

```
twice(_ + 1, 5)  // returns 7.0
```

# Currying & New Control Structures

More useful example: **Loan pattern** technique.

Informally, the loan pattern could be described as follows:

- **Open a resource**
- **Perform operations on the resource**
- **Close the resource**

The advantage of using this control pattern is that it composes actions related to a resource in the single structure. So it's impossible for the client of the control structure to forget to close the resource previously opened.

# Currying & New Control Structures

More useful example: **Loan pattern** technique.

```
def printing ( file: File, printer: PrintWriter => Unit)
{
  val writer = new PrintWriter(file) // open resource
  try {
    printer(writer)
  }
  finally {
    writer.close()
       // close resource
  }
}
```

The advantage of using this method is that it's `print`, not user code, that assures the file is closed at the end. So it's impossible for user to forget to close the file.

How to use `printing`:

```
printing (
  new File("data.txt"),
  writer => writer.println(new java.util.Date)
)
```

# Currying & New Control Structures

How to make printing look like a usual control structure?

**First step**: make it curried:

```
def printing (file: File)(printer: PrintWriter => Unit)
{
  val writer = new PrintWriter(file) // open resource
  try {
    printer(writer)
  }
  finally {
    writer.close()
       // close resource
  }
}
```

How to use printing:

```
printing(new File("data.txt"))
(
  writer => writer.println(new java.util.Date)
)
```

# Currying & New Control Structures

How to make `printing` look like a usual control structure?
**Second step**: Use the following rule:

> In any method invocation in which you are passing **exactly one** argument, you can use **curly braces** to surround the argument **instead of parentheses**.

How to use `printing` (the final result):

```
printing(new File("data.txt"))
{
  writer => writer.println(new java.util.Date)
}
```

# Assignment

Write the implementation of the **loan pattern** as a "new" control structure. The structure should repeatedly perform the same action under some condition – something like while loop.

For that, prepare a curried function like f(*condition*)(*action*), and, perhaps, use the mechanism of by-name parameters.

Provide a reasonable example of using the control structure.